REMARKS

The present amendment is in response to the Examiner's action of March 25, 2004. A petition for extension of time and the necessary fees are enclosed.

Claims 1-4 have been amended to better describe the present invention. Claims 8-11 have been added to recite additional aspects of the present invention.

35 U.S.C. § 112 Rejections

The examiner rejected claims 1, 2, 4, 6, 7 as indefinite under 35 U.S.C. § 112.

Referring to claim 1, the examiner stated that the claim is indefinite because several aspects of it are unclear. Claim 1, was amended to improve its clarity and to recite better the relationship between the various elements. Applicants respectfully submit that as amended claim 1 is not ambiguous.

The Examiner states that it is not clear "how the steps of claim 1 allow arbitrary protocols to be added or plugged into a middleware based application without accessing the source code" (Examiner's action, pg. 2, last paragraph). Applicants assert that claim 1, as amended and when read in view of the specification, shows that arbitrary protocols may be plugged into middleware by using a connection bridge as an intermediary. Therefore, the source code of the middleware need not be accessed in order to allow it to interface with arbitrary protocols, as the middleware interacts with the protocols via the connection bridge. Thus, the connection bridge can present a single predefined interface to the middleware regardless of which protocol is attached to the connection bridge (see generally page 27, lines 3-20).

The Examiner further states that it is unclear how a protocol can perform actions, as the protocol is merely a set of rules (*Id.*) The Examiner relies on one of several possible definitions of the term "protocol". While the Examiner correctly refers to the classical definition of "protocol" as a set of rules, in modern usage in the art the term "protocol" has obtained an additional related meaning. That meaning is a software or hardware utility or service that implements a set of rules that govern how two or more applications or devices communicate. A person skilled in the art can usually determine which definition of the term "protocol" is intended by the context in which the term is used. Claim 1 recites that a transport protocol performs specific actions, such as generating an action request, and sending an action request. Therefore, a person skilled in the art would determine that the intended definition of protocol is that of a utility implementing a set of rules. See page 927 of The Computer Desktop Encyclopedia, 2nd Edition, The Computer Language Company (1999), copy enclosed and referred to hereinafter as the "Encyclopedia.", which defines a transport protocol as:

> a communications protocol responsible for establishing a connection and ensuring that all data has arrived safely. It is defined in layer 4 of the OSI model. Often the term transport protocol implies transport services, which includes the lower level data link protocol that moves packets form one node to another.

Thus, there are no ambiguities as to how such a utility may communicate and generate an action request.

The Examiner further states that "it is unclear ... what the action request is requesting and where the action will be performed" (*Id.*) Applicant respectfully asserts that there is no ambiguity when the claim is read in the context of the specification. Action requests are described in the specification (see e.g., page 23 line 10 through page 24 line 9). Thus, "the middleware ... performs the action" (page 23, lines 10, 11); also see "the middleware then will perform the action requested" (page 37, lines 18, 19). The specification further describes several examples of action

requests (page 24, lines 1-9). It can be seen that action requests are requests for operations necessary for managing connection-based communications, such as accepting a connection, and invoking a process. For the above described reasons, it is respectfully submitted that claim 1 is not indefinite.

Claim 2 is amended to improve the clarity of the claim. Applicant respectfully asserts that as amended, claim 2 is not indefinite.

Claim 4 is also amended to improve the clarity of the claim. Applicant respectfully asserts that as amended, claim 4 is not indefinite.

Referring to claim 6, the Examiner states that the term "utilizing a synchronization primitive" is unclear (Examiner's action, page 3, paragraph 3). Applicants respectfully disagree. A primitive is known in the art to be a function which is built into a programming language or operating system, either for speed of execution or because it would be impossible to implement it otherwise. See the Encyclopedia page 724. Referring to the specification (e.g. page 27, line 9; page 28, line 7) and the context of the present invention (the present invention is not directed to programming languages), it is clear that the claim refers to a primitive which is built into the operating system. A synchronization primitive is a known type of primitive that handles synchronization between various processes in an operating system. An example of a synchronization primitive is the semaphore (see page 28, line 8). According to the standard definition, a semaphore blocks a process from executing until the semaphore has reached a certain predefined value. In the present invention a semaphore is used for notification (see page 27, lines 9-11 and page 28, lines 6-9). A person skilled in the art will realize that utilizing a semaphore for notification means that the semaphore is used to block a process, and the notification is realized by allowing the process to run by changing the semaphore's value. Other synchronization primitives may be used in similar manner.

Claim 7 was rejected for lack of antecedent basis. It is believed that claim 7 is patentable as amended.

Reisman (U.S. Patent No. 6,611,862) Rejections

The Examiner rejected independent claim 1 as anticipated by Reisman (U.S. Patent No. 6,611,862). Applicants respectfully disagree.

Reisman describes a user station that connects to various consumer information sources (such as newspapers, magazines, etc.), sends user data from these sources and retrieves information which is presented to the user.

However, Reisman does not disclose a distributed application as recited by claim 1. Reisman discloses multiple targeted online services and a client interface, which are two completely distinct systems (col. 24, lines 48, 49). Reisman further discloses a user station communicating with independently operated data sources (col. 5, lines 15-17). If the data sources are independently operated they are not part of a single distributed application as required by claim 1. Since Reisman does not disclose a distributed application, it does not disclose a first application software, and a second application software, wherein both application softwares are part of the same distributed application.

Furthermore, Reisman does not disclose middleware, as recited in claim 1. As defined by the specification, middleware is a sub-system used to handle communications for a distributed computing system and thus hide the operating system and network transport protocol programming needed to realize such a system. Reisman does not disclose middleware, because he does not disclose a distributed system.

In addition, Reisman does not disclose transport protocols, as recited by claim 1. Computer communication protocols are usually grouped on several levels (or layers) according to the tasks of each protocol. The most well known (and widely used) of these groupings is the Open System Interconnect (OSI) model which groups protocols in 7 layers. Each layer relies on functions provided by the layer below it. Thus, lower layers deal with certain aspects of the communication system which are invisible to higher layers. Therefore, systems used for dealing with higher level protocols are usually unusable for lower level protocols, as lower level protocols require features that are not present in the higher layers. For example, the transport layer (OSI layer 4) handles connections between two points and ensures that messages can travel between these two points error free and in the correct order. Therefore, layers above the transport layer, such as the application layer (layer 7) do not handle connections. A system or software created for the application layer cannot be readily used at the transport layer, because it is not configured to handle the functions of the transport layer.

The text cited by Examiner (col. 24, lines 48-58) shows that Reisman teaches the handling of application protocols, not transport protocols. Thus, when Reisman states that "modules 88 mimics the online services protocols" (col. 24, lines 53-54) it is referring to application protocols. This is the case because the online services are end user applications, and not communications utilities (see col. 2, lines 20-67). For the reasons described above, a system based on the application layer, cannot be easily used at the transport layer. Referring to claim 1, the transport layer protocols require management of connections, and thus handling of action requests and connection identifiers. This is not disclosed by Reisman, as Reisman does not teach direct interaction with transport layer protocols.

In reference to claims 6 and 7, Reisman does not teach utilizing primitives. In reference to claim 8, Resiman does not teach communicating executable code, only information. In reference to claim 11, Reisman does not teach an embedded computer.

Reisman does not render claim 1 obvious. It is not obvious to modify a system for browsing information (as disclosed by Reisman), into a system for distributed computing. Furthermore, Reisman does not directly interact with transport protocols and therefore does not disclose the complex features necessary for direct communication with transport protocols, such as management of connections by using action requests and connection identifiers. Adding these features to the system described by Reisman is not obvious, as it would require a substantial engineering effort. Furthermore, Reisman does not include any suggestion to undertake such modifications.

Claims 2-11 are patentable over Reisman because they depend from claim 1, which is patentable.

Ben-Shachar (U.S. Patent No. 6,209,018) rejections

The examiner also rejected independent claim 1 as anticipated by Ben-Shachar (U.S. Patent No. 6,209,018). Applicants respectfully disagree.

While Ben-Shachar does disclose a distributed system and middleware, it is directed to an entirely different endeavor than the one to which the present invention is directed. Ben-Shachar is a system for efficiently handling, distributing and processing service requests to a server. While he does use communication protocols, Ben-Shachar is not concerned with whether the communication protocols are compatible with the middleware (i.e., CORBA). Thus, while Ben-Shachar discloses using middleware and communications protocols, it does not teach any improvements upon them. Accordingly, Ben-Shachar does not disclose adding support for communication protocols to middleware without accessing the source code of the middleware.

The text cited by the Examiner (col. 9, lines 27-43) does not disclose most of the elements of claim 1, such as, for example one or more transport protocols, a connection bridge, generating an action request, a protocol connection identifier, sending the action request, notifying the middleware, and transferring the action request to the middleware.

Applicant is not certain which particular portions of the cited text are alleged by the Examiner as anticipating each of the above elements. Therefore, Applicant cannot offer specific responses. However, it should be noted that despite including the term "plug-in", the "CGI, NSAPI, ISAPI plug-in[s]" (col. 9, line 29) are not comparable to the transport protocols or the connection bridge of claim 1. These plug-ins are essentially software applications that run on a server. They use CORBA (i.e., the middleware) for communication, while the transport protocols and the communication bridge, as recited in claim 1, are used by CORBA for communication. Thus, Ben-Shachar states that "the plug-in represents a CORBA client" (col. 9, lines 30, 31), which indicates that the plug-in merely uses the middleware, and does not allow it to support additional protocols. The term "plug-in" as used in this passage of Ben-Shachar does not mean a plug-in to CORBA, but a plug-in to a web-service.

Claim 1 is not rendered obvious by Ben-Shachar, because as shown above Ben-Shachar is directed to a entirely different system. Therefore, Claim 1 is patentable over Ben-Shachar. Claims 2-11 are patentable because they depend from claim 1, which is patentable.

Conclusion

The Examiner did not attempt to combine Reisman and Ben-Shachar. Applicant asserts that there is no suggestion to combine Resiman and Ben-Shachar as they are directed to different systems. While Reisman is directed to client computers, Ben-Shachar is directed to servers and services. Therefore, even if they were combined, the features of Reisman and Ben-Shachar would

reside at different computers, and therefore they could not be applied together to invalidate the elements of claim 1, the majority of which reside on a single computer (the first computer).

Even if combined, Reisman and Ben-Shachar would not disclose many of the elements of claim 1, such as, for example, one or more transport protocols, an action request, and a protocol connection identifier.

In view of the above, each of the presently pending claims in this application is believed to be in immediate condition for allowance. Accordingly, the Examiner is respectfully requested to pass this application to issue.

Dated: July 26, 2004

Respectfully submitted,

By _____
Hristo I. Vachovsky
Registration No.: 55,694
DARBY & DARBY P.C.
P.O. Box 5257
New York, New York 10150-5257
(212) 527-7700
(212) 753-6237 (Fax)
Attorneys/Agents For Applicant